ThreatConnect® Upgrade Guide: Containerized Deployment

Software Version 7.10.2

Technical Guide

October 1, 2025

10033-13 EN Rev. A

ThreatConnect, Inc.
3865 Wilson Blvd., Suite 550, Arlington, VA 22203

P: 1.800.965.2708 | F: 703.229.4489
www.ThreatConnect.com

Table of Contents

Overview **5**

Upgrade Steps **5**

Step 1: Upgrade Docker Files 5

Step 2: Fix Shell Scripts 6

Step 3: Configuration Updates 6

Upgrade From 7.10.x to 7.10.2 6

Upgrade From pre-7.10 Versions to 7.10.2 8

Step 4: Log Into ThreatConnects ECR 8

Step 5: Stop and Remove Containers 9

Step 6: Export Environment Variables 9

Step 7: Start ThreatConnect 9

Start OpenSearch 10

Start Postgres 10

Start Redis 10

Start tc-mon 11

Start tc-app 12

Start tc-job 12

Step 8: Restart and Monitor ThreatConnect 12

Step 9: Unset Environment Variables 13

Step 10: Re-create the Search Index 13

Step 11: Update the Search Index 14

Step 12: Clean Up Old Container Images 15

Appendix **16**

Enable SAML 16

Configure TLS in the Postgres Container 16

Configure TLS in the Redis Container 17

Change the UID for the Postgres Container 18

Change the UID for the Redis Container 20

Change the UID for the ThreatConnect Containers 21

Add Certificates 22

Certificate Authority Signed Certificate 23

Self-Signed Certificate 25

Changelog 26

7.10.2 Changes 26
7.10.0 Changes 26
Cipher Suites Settings 27
TLS Connection to Containers 27
Trusted Certificates 27
Use Any UID 27
Redis Connection Settings Variables 28
Preserve MDB Settings 29
AWS Document Storage 29
Remove Nginx Container 30

Overview

This guide describes how to upgrade ThreatConnect® on a ThreatConnect instance that is running in a containerized solution using Docker® or Podman.

Important: The containerized deployment was tested on AlmaLinux OS™ 9 and Red Hat® Enterprise Linux® (RHEL) 8 and 9 and is the standard deployment method for all production and non-production systems. For instructions on installing ThreatConnect and having it run directly on an operating system (OS), see *ThreatConnect Installation Guide: Linux Operating System Legacy Deployment*.

Important: Instances running on Red Hat® Enterprise Linux® (RHEL) 8 must use Podman for containerized deployments. Rootless Podman is recommended.

Upgrade Steps

Important: All of the steps in this guide apply to all Docker and Podman deployments.

Important: For Podman deployments, the rest of the steps in this guide assume the user executing them is the user currently running the containers.

Step 1: Upgrade Docker Files

Important: This step applies to Docker *and* Podman deployments. The term "Docker" in the ZIP file is just part of the filename and is not specific to Docker deployments.

Note: You must complete this step on all hosts that run ThreatConnect or some component of ThreatConnect.

Important: If you have changed any of the following files locally, you must back them up, because they will get overwritten during the upgrade:

- `opensearch_internal_users.yml`
- `pg_hba.conf`
- `postgres.conf`
- `redis.conf`

Follow these steps to upgrade the Docker files:

1. Upgrade the ThreatConnect Docker ZIP file to the latest version (replace the `<version number>` placeholder value with the version number for the ThreatConnect version to which you are upgrading):

```
None
unzip -o threatconnect-docker-v<version number>.zip
cd threatconnect-docker
```

Important: The rest of the steps in this guide assume you are in the unzipped ThreatConnect Docker directory.

2. Redo any local changes that you had in the following files, taking care not to overwrite incoming updates from ThreatConnect:
    - `opensearch_internal_users.yml`
    - `pg_hba.conf`
    - `postgres.conf`
    - `redis.conf`

Step 2: Fix Shell Scripts

Note: You must complete this step on all hosts that run ThreatConnect or some component of ThreatConnect.

Reformat and change permissions on shell scripts:

```
None
sed -i 's/\r$//' fix_shell_scripts.sh
chmod 755 fix_shell_scripts.sh
./fix_shell_scripts.sh
```

Step 3: Configuration Updates

Upgrade From 7.10.*x* to 7.10.2

Follow these steps if you are upgrading from version 7.10.*x* to 7.10.2:

1. Back up the `.env` file:

```
None
# backup .env if exists
cp .env .env.$(date +%Y%m%d%H%M%S)
```

2. Remove the following sensitive variables from the `.env` file:

```
None
DB_PASS=
DB_SUPER_USER_PASS=
OPENSEARCH_PASSWORD=
REDIS_PASS=
SAML_KEYSTORE_PASS=
TC_KEYSTORE_PASS=
TC_SMTP_PASS=
```

3. Remove the `CUSTOM_CA` environment variable from the `.env` file. For custom or non-public CA-signed certificates, simply add the `ca.pem` file to `certs/trusted`.
4. OpenSearch®, Postgres®, and Redis® must be upgraded to fix vulnerabilities. Make the following updates in your `.env` file:

```
None
OPENSEARCH_VERSION=3.2.0
POSTGRES_VERSION=14.19
REDIS_VERSION=8.2.1
```

5. Add the following to the `.env` file:

```
None
# OpenSearch Cluster name [Required].
OPENSEARCH_CLUSTER=search-cluster
# Enter SAML2's skew milliseconds. Represented in threatconnect.xml as
sso.logout.interval.seconds
SAML_LOGOUT_INTERVAL_SECONDS_VALUE=600
```

Upgrade From pre-7.10 Versions to 7.10.2

Follow these steps if you are upgrading from pre-7.10 versions (that is, version 7.9.*x* or earlier) to 7.10.2:

1. Back up the `.env` file if one exists:

```
None
cp .env .env.$(date +%Y%m%d%H%M%S)
cp .env.sample .env
```

2. Update each variable in the `.env` file with the appropriate value. For descriptions of the values, refer to the comments in that file.

3. Fix the `.env` file and source it to your environment:

```
None
sed -i 's/\r$//' .env
source .env
```

Step 4: Log Into ThreatConnects ECR

Log into ThreatConnect's Elastic® Container Registry (ECR):

Important: If your system is located in a time zone other than `us-east-1`, you can replace `us-east-1` with a different Amazon Web Services® (AWS) region before running these commands. The following AWS regions are supported at this time: `ap-northeast-1`, `ap-southeast-2`, `ca-central-1`, `eu-central-1`, `eu-north-1`, `eu-south-1`, `eu-west-2`, `me-south-1`, `sa-east-1`, `us-east-1`, `us-east-2`, `us-west-1`, `us-west-2`.

```
None
docker login \
  -u AWS \
  -p $(/usr/local/bin/aws ecr get-login-password --region us-east-1) \
  373319941383.dkr.ecr.us-east-1.amazonaws.com
```

Step 5: Stop and Remove Containers

Note: You must complete this step on all hosts running any component of ThreatConnect.

Run the following command to stop and remove all containers:

```
None
docker-compose down
```

Step 6: Export Environment Variables

Note: You must complete this step on all hosts running a ThreatConnect server (**tc-mon**, **tc-app**, and **tc-job**).

Run the following commands to export all environment variables, replacing `<DB_PASS>`, `<DB_SUPER_USER_PASS>`, `<OPENSEARCH_PASSWORD>`, `<REDIS_PASS>`, and `<TC_SMTP_PASS>` with the appropriate values:

```
None
export DB_PASS=<DB_PASS>
export DB_SUPER_USER_PASS=<DB_SUPER_USER_PASS>
export OPENSEARCH_PASSWORD=<OPENSEARCH_PASSWORD>
export REDIS_PASS=<REDIS_PASS>
export TC_SMTP_PASS=<TC_SMTP_PASS>
```

Step 7: Start ThreatConnect

Start each of the following services in the following order: OpenSearch → Postgres → Redis → **tc-mon** → **tc-app** → **tc-job**. After starting each service, make sure to perform the following actions:

- Run `docker-compose logs --tail=10 --follow` to verify the service starts up before moving on to the next.
- Press **Ctrl+C** once the service is started.

Start OpenSearch

Note: You must complete this step on the host running OpenSearch.

Follow these steps to start OpenSearch:

1. Export the current OpenSearch password, replacing `<OPENSEARCH_PASSWORD>` with the appropriate value, and then start OpenSearch:

```
None
export OPENSEARCH_PASSWORD=<OPENSEARCH_PASSWORD>
docker-compose up -d opensearch
```

2. Set the OpenSearch password:

```
None
./set_opensearch_password.sh
```

Start Postgres

Note: You must complete this step on the host running Postgres.

Export the current Postgres password, replacing `<DB_SUPER_USER_PASS>` with the appropriate value, and then start Postgres:

```
None
export DB_SUPER_USER_PASS=<DB_SUPER_USER_PASS>
docker-compose up -d postgres
```

Start Redis

Note: You must complete this step on the host running Redis.

Follow these steps to start Redis:

1. Start Redis:

```
None
docker-compose up -d redis
```

2. Test Redis:
    a. Navigate to a command prompt in the `redis` container:

```
None
docker exec -ti redis bash
```

    b. Use either Option 1 or Option 2, depending on your configuration:
        - **Option 1**: Execute this command if you are using non-secure Redis (default):

```
None
redis-cli
```

        - **Option 2**: Execute this command if you are using Secure Redis, replacing `<REDIS_USER>` and `<REDIS_PASS>` with the appropriate values:

```
None
redis-cli --tls --cacert /certs/cert.pem --cert /certs/cert.pem --key
/certs/privkey.pem
auth <REDIS_USER> <REDIS_PASS>
```

    c. Ping the redis server (expected result: `PONG`):

```
None
ping
```

Start tc-mon

Note: You must complete this step on the host running the ThreatConnect messaging server.

Run the following command to start **tc-mon**:

```
None
docker-compose up -d tc-mon
```

Start tc-app

Note: You must complete this step on the host running the ThreatConnect application server.

Run the following command to start **tc-app**:

```
None
docker-compose up -d tc-app
```

Start tc-job

Note: You must complete this step on the host running the ThreatConnect Playbooks server.

Run the following command to start **tc-job**:

```
None
docker-compose up -d tc-job
```

Step 8: Restart and Monitor ThreatConnect

Follow these steps to restart and monitor the ThreatConnect containers:

1. Run the following command to check the status of the ThreatConnect containers.

```
None
docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Status}}"
```

2. If you need to restart the ThreatConnect containers without modifying them or setting environment variables, run the following commands one at a time, waiting for each container to start (about 60-90 seconds) before entering the next command:

```
None
docker restart tc-mon
docker restart tc-app
docker restart tc-job
```

3. If environment variables for `TC_APP_LOGS`, `TC_JOB_LOGS`, and `TC_MON_LOGS` are set, then run the following commands to tail monitor the ThreatConnect logs, respectively:

```
None
tail -f $TC_APP_LOGS/server.log $TC_APP_LOGS/tc.log
tail -f $TC_JOB_LOGS/server.log $TC_JOB_LOGS/tc.log
tail -f $TC_MON_LOGS/server.log $TC_MON_LOGS/tc.log
```

ThreatConnect logs can also be found in the following locations:

- Docker: `/var/lib/docker/volumes/`
- Podman: `$XDG_DATA_HOME/containers/storage/`

Step 9: Unset Environment Variables

Unset the following sensitive environment variables:

```
None
unset DB_PASS
unset DB_SUPER_USER_PASS
unset OPENSEARCH_PASSWORD
unset REDIS_PASS
unset TC_SMTP_PASS
```

Important: If your system is Security Assertion Markup Language™ (SAML™) enabled, then you must also unset `SAML_KEYSTORE_PASS`.

Step 10: Re-create the Search Index

Important: If you are **upgrading from ThreatConnect 7.7.0 or later**, skip this step.

Follow these steps to re-create the search index:

1. Log into ThreatConnect with a System Administrator account.
2. From the **Settings** menu on the top navigation bar, select **System Settings**.
3. Click **CREATE SEARCH INDEX** at the top right of the **Settings** tab.
4. On the **Setup** tab of the **Search Index Configuration** window, select the **Perform search indexing on database source** and **Load file contents into search index** checkboxes, and then click **INITIALIZE**.

   Note: The **Perform search indexing on database source** checkbox determines whether to index objects that exist in the ThreatConnect database, and the **Load file contents into search index** checkbox determines whether to index objects that exist in document storage.

Step 11: Update the Search Index

Important: If you are **upgrading from ThreatConnect 7.7.0 or later**, skip this step.

Update the search index on the host that is running the OpenSearch Docker container (replace the `<opensearch username>` and `<opensearch password>` placeholder values):

```
None
curl -sk -XPOST
"https://$OPENSEARCH_USER:$OPENSEARCH_PASS@localhost:9200/orgs/_close"
curl -sk -XPUT
"https://$OPENSEARCH_USER:$OPENSEARCH_PASS@localhost:9200/orgs/_settings" -H
'Content-Type: application/json' -d'
{
  "analysis.analyzer": {
    "default_search": {
      "filter": [
        "lowercase",
        "asciifolding"
      ],
      "type": "custom",
      "tokenizer": "whitespace"
    }
```

```
  }
}'
curl -sk -XPOST
"https://$OPENSEARCH_USER:$OPENSEARCH_PASS@localhost:9200/orgs/_open"
```

Step 12: Clean Up Old Container Images

Important: It is important to remove old container images because they are several gigabytes in size.

Run the following commands to clean up old container images, replacing `<old container image id>` with the image ID:

```
None
docker image list
docker image rm <old container image ID>
```

Appendix

Enable SAML

Follow these steps to enable the SAMLconfiguration on ThreatConnect:

1. In the `.env` file associated with the containerized deployment of ThreatConnect, update each variable in the "SAML Settings" section with the appropriate value. For descriptions of the values that you must provide in the `.env` file, reference the comments in the "SAML Settings" section of that file.
2. Add the following `.pem` files to the `certs` folder:
    - `certs/saml_privkey.pem`
    - `certs/saml_fullchain.pem`
    - `certs/saml_host.pem`

    Note: The `saml_fullchain.pem` and `saml_privkey.pem` files can have the same content as the `fullchain.pem` and `privkey.pem` files. The `saml_host.pem` file must contain the Identity Provider (IDP) certificate.

Configure TLS in the Postgres Container

Follow these steps to configure transport layer security (TLS) in the Postgres container:

1. Remove empty folders that may have been created in the absence of certificate files (if applicable):

```
None
rm -rf certs/fullchain.pem/ certs/privkey.pem/ certs/postgres-privkey.pem/
```

2. Follow the steps in Add Certificates to add certificates on the host running the Postgres container.
3. Find the UID that Postgres uses:
    - Docker:

```
None
ls -l /var/lib/docker/volumes/threatconnect-docker_postgres-data
```

- Podman:

```
None
ls -l run/containers/storage/volumes/threatconnect-docker_postgres
```

4. Run the following command to change ownership of `postgres-privkey.pem` to the user the Postgres container uses (replace `<pg uid>` with the UID found in Step 3):

   Note: For rootless Podman deployments, this command must be executed as root.

```
None
chown <pg uid>:<pg uid> certs/postgres-privkey.pem
```

5. Uncomment the TLS settings in `config/postgres.conf`.
6. Uncomment the last two lines to force SSL in `config/pg_hba.conf`.
7. Restart Postgres:

```
None
docker rm -f postgres && docker-compose up -d postgres
```

8. Restart the ThreatConnect containers.

Configure TLS in the Redis Container

1. Follow the steps in Add Certificates to add certificates on the host running the Redis container.
2. Uncomment the TLS settings in `config/redis.conf`.
3. Update the following environment variables (replace `<redis user>` and `<redis pass>` with same values entered on the last line of `redis.conf`):

```
None
# Secure Redis. [Optional] [true | false]
REDIS_SECURE=true
# Secure Redis User Name. [Optional]
REDIS_USER=<redis user>
```

```
# Secure Redis Password. [Optional]
REDIS_PASS=<redis pass>
# Secure Redis TLS. [Optional] [true | false]
REDIS_TLS=true
```

4. Restart Redis:

```
None
docker rm -f redis && docker-compose up -d redis
```

5. Restart the ThreatConnect containers.

Change the UID for the Postgres Container

Warning: Changing the UID of an already built Postgres container requires creating a new database and migrating the data.

Follow these steps to change the UID for the Postgres container:

1. Create a dump file of the `threatconnect` database, and then shut down Postgres:

```
None
source .env
/usr/bin/docker-compose exec -e PGPASSWORD=${DB_SUPER_USER_PASS} postgres
pg_dump -U $DB_SUPER_USER threatconnect > threatconnect.sql
docker rm -f postgres
```

2. Set values for the following environment variables:

```
None
# The UID that must exist on this host to be used by the Database container
[Required].
DB_UID=
# The GID that must exist on this host to be used by the Database container
[Required].
DB_GID=
```

```
# Postgres data folder [Required if DB_UID is set]. Set an absolute path, owned
by user DB_UID. (Ex. /opt/threatconnect-docker/postgres-data)
POSTGRES_DATA=
```

3. Create the user and Postgres data folders:

   Important: The username can be anything. This guide uses the name `postgres`.

```
None
source .env
groupadd -g $DB_GID postgres
adduser --uid $DB_UID --gid $DB_GID postgres
mkdir $POSTGRES_DATA
chown $DB_UID:$DB_GID -R $POSTGRES_DATA
```

4. Change ownership of `postgres-privkey.pem`:

```
None
chown $DB_UID:$DB_GID certs/postgres-privkey.pem
```

5. Unshare the Postgres data folder and `postgres-privkey.pem`:

   Note: This step applies to Podman deployments only. For Docker deployments, skip to Step 6.

```
None
docker unshare chown ${DB_UID}:${DB_GID} -R ${POSTGRES_DATA}
docker unshare chown ${DB_UID}:${DB_GID} certs/postgres-privkey.pem
```

6. Start Postgres:

   Note: This step creates a new database.

```
None
docker-compose up -d postgres
```

7. Copy the dump file to the `schema` folder:

```
None
mv threatconnect.sql schema/
```

8. Create the db user and role:

```
None
bash create_db_user.sh
```

9. Ingest the dump file:

   Note: This process may take a while, depending on the size of the dump file.

```
None
/usr/bin/docker-compose exec -e PGPASSWORD=${DB_SUPER_USER_PASS} postgres psql
-U $DB_SUPER_USER -d $DB_NAME -f /schema/threatconnect.sql
```

10. Restart the ThreatConnect containers.

Change the UID for the Redis Container

Follow these steps to change the UID for the Redis container:

1. Set values for the following environment variables:

```
None
# Redis user UID
REDIS_UID=
# Redis user GID
REDIS_GID=
# Redis data folder [Required if REDIS_UID is set]. Set an absolute path, owned
by user REDIS_UID (Ex. /opt/threatconnect-docker/redis-data)
REDIS_DATA=
```

2. Create the user and Redis data folder:

   Important: The username can be anything. This guide uses the name `redis`.

```
None
source .env
groupadd -g $REDIS_GID redis
adduser --uid $REDIS_UID --gid $REDIS_GID redis
mkdir -p ${REDIS_DATA}
chown ${REDIS_UID}:${REDIS_GID} -R ${REDIS_DATA}
```

3. Unshare chown the Redis data folder:

   Note: This step applies to Podman deployments only. For Docker deployments, skip to Step 4.

```
None
docker unshare chown ${REDIS_UID}:${REDIS_GID} -R ${REDIS_DATA}
```

4. Restart Redis:

```
None
docker rm -f redis && docker-compose up -d redis
```

5. Restart the ThreatConnect containers.

Change the UID for the ThreatConnect Containers

Note: This procedure applies to Docker deployments only.

Warning: Changing the UID of an already built ThreatConnect container requires changing ownership of all files and folders owned by it.

1. Set values for the following environment variables:

```
None
# The main UID that must exist on this host to be used by ThreatConnect
containers [Required].
THREATCONNECT_UID=
# The main GID which must exist on this host to be used by ThreatConnect
containers [Required].
THREATCONNECT_GID=
```

```
# The tc-job UID that must exist on this host to be used by tc-mon, tc-app, and
tc-job containers [Required].
TC_JOB_UID=
# The GID that must exist on this host to be used by tc-mon, tc-app, and tc-job
containers [Required].
TC_JOB_GID=
```

2. Create user accounts:

   Important: The usernames can be anything. This guide uses the names `threatconnect` and `tc-job`.

```
None
source .env
groupadd -g $THREATCONNECT_GID threatconnect
adduser --uid $THREATCONNECT_UID --gid $THREATCONNECT_GID threatconnect
adduser --uid $TC_JOB_UID tc-job
```

3. Change ownership of the doc storage folder:

   Note: This process may take a while, depending on the size of the folder.

```
None
chown $THREATCONNECT_UID:$THREATCONNECT_GID -R ${TC_DOC_STORAGE}
```

4. Restart the ThreatConnect containers.

   Note: This process may take a while, because it finds and changes ownership of every file and folder.

Add Certificates

Follow these steps to add certificates:

1. Obtain the required certificates. These are the certificate authority (CA) certificate (`ca.pem`), the CA-signed certificate (`server.pem`), and the private key (`privkey.pem`). Copy them to the `certs/` folder. If you do not have a signed certificate bundle, refer to the following steps:

- Certificate Authority Signed Certificate
- Self-Signed Certificate

2. Concatenate `server.pem`, `ca.pem`, and any provided intermediate certificates (e.g., `intermediate.pem`) to a file called `fullchain.pem`:

```
None
cp /path/to/server.pem certs/fullchain.pem
cat /path/to/intermediate.pem >> certs/fullchain.pem
cat /path/to/ca.pem >> certs/fullchain.pem
cp /path/to/privkey.pem certs/privkey.pem
cp certs/privkey.pem certs/postgres-privkey.pem
```

3. Create the trusted directory:

```
None
mkdir certs/trusted
```

4. Copy pem-formatted certificates of any root CAs you wish ThreatConnect to trust, such as in the following example:

   Note: Any certificates added in this step will automatically be added to the environments of all ThreatConnect Playbooks, Applications, and Services.

```
None
cp ~/my.trusted.ca.pem certs/trusted/
```

5. Correct ownership and permissions on certificate files:

```
None
chmod 744 -R certs
chmod 644 certs/privkey.pem
chmod 600 certs/postgres-privkey.pem
```

Certificate Authority Signed Certificate

If you do not have a server or certificate authority (CA) certificate, follow these steps to generate a certificate request:

1. Create the `privkey.pem`:

```
None
mkdir certs && cd certs
openssl genrsa -out privkey.pem 4096
```

2. Create the Certificate Signing Request Configuration File (`certreq.cnf`):

   Note: Replace all values surrounded with `<>` with real values.

```
None
[req]
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no
[req_distinguished_name]
CN = <The fully qualified domain name (FQDN) or IP address of the server>
OU = <The name of the organizational unit>
O = <The name of the organization>
L = <The location or city in which the organization resides>
S = <The state or province in which the organization resides>
[req_ext]
subjectAltName = @alt_names
[alt_names]
IP.1 = <The IP address of the server>
DNS.1 = <The FQDN of the server>
```

3. Create the Certificate Signing Request (`certreq.csr`):

```
None
openssl req -new -sha256 -key privkey.pem -out certreq.csr -config certreq.cnf
```

4. Submit `certreq.csr` to your CA, who will return a signed certificate bundle. If possible, specify PEM as the return format.
5. If your CA is not public, then update `CUSTOM_CA` in your `.env` file as follows:

```
None
CUSTOM_CA=true
```

Self-Signed Certificate

If you do not have a server or CA certificate, follow these steps to generate self-signed certificates:

1. Create a certificate authority (replace the `<country>`, `<state>`, `<city>`, `<company>`, and `<department>` placeholder values):

```
None
mkdir certs && cd certs
openssl genrsa -out ca-key.pem 4096
openssl req -new -x509 -sha256 -key ca-key.pem \
    -subj "/C=<country>/ST=<state>/L=<city>/O=<company>/OU=<department>/CN=My
Root Authority" \
    -out ca.pem -days 3650
```

2. Create a private key:

```
None
openssl genrsa -out privkey.pem 4096
```

3. Create a certificate signing request (replace the `<country>`, `<state>`, `<city>`, `<company>`, `<department>`, and `<FQDN/IP of server>` placeholder values):

```
None
openssl req -new -sha256 -key privkey.pem \
    -subj
"/C=<country>/ST=<state>/L=<city>/O=<company>/OU=<department>/CN=<FQDN/IP of
server>" \
    -out <FQDN/IP of Server>.csr
```

4. Create a new file for alternate names in `alt-names.ext` (replace the `<FQDN/IP of server>` placeholder value):

```
None
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
```

```
subjectAltName = @alt_names

[alt_names]
DNS.1 = <FQDN/IP of server>
```

5. Create a certificate signed by your CA (replace the `<FQDN/IP of server>` placeholder value):

```
None
openssl x509 -req -in <FQDN/IP of server>.csr -CA ca.pem \
    -CAkey ca-key.pem -CAcreateserial \
    -out server.pem -days 398 -sha256 \
    -extfile alt-names.ext
```

6. Update `CUSTOM_CA` in your `.env` file as follows:

```
None
CUSTOM_CA=true
```

Changelog

7.10.2 Changes

- Fixed vulnerabilities in Opensearch, Postgres, and Redis containers.
- The **JS Report** Service was fixed to allow it to work in a fully distributed ThreatConnect environment where the Redis server is secured with certificate-based TLS encryption, using a custom certificate authority (CA) signed certificate.
- All plain-text passwords have been removed from the `.env` file and should be set as environment variables instead.

7.10.0 Changes

- Added cipher suites settings
- Added support for TLS connection to Postgres and Redis containers
- Added support for trusted certificates

- Added support for using any UID
- Added variables for Redis connection settings
- Added variable to preserve MDB settings
- Added support for AWS Document Storage
- Removed the nginx container

Cipher Suites Settings

The following environment variables were added to control the cipher suites ThreatConnect uses in both its https connections and messaging (JMS):

```
None
# The ciphersuites the messaging server will use to secure communications [Optional]. This must be a comma delimited string that contains TLS1.2 and/or TLS1.3 ciphersuite names. Leave blank to use default ciphers.
TC_MESSAGING_CIPHERS=
# The TLS1.2 ciphersuites to secure communications [Optional]. This must be a comma or colon delimited string containing TLS1.2 ciphersuite names only. Leave blank to use default ciphers.
TC_TLS2_CIPHERS=
# The TLS1.3 ciphersuites to secure communications [Optional]. This must be a colon delimited string containing TLS1.3 ciphersuite names only. Leave blank to use default ciphers.
TC_TLS3_CIPHERS=
```

TLS Connection to Containers

Support for enabling TLS connections with customer certificates is now available for the Postgres and Redis containers. See Configure TLS in the Postgres Container and Configure TLS in the Redis Container for details.

Trusted Certificates

You can now drop Apps into the `certs/trusted` folder to make the Apps automatically trust external endpoints that are protected by private or self-signed certificates. The certificates must be in PEM (`.pem`) format.

Use Any UID

The following environment variables were added to enable you to use any UID:

Warning: Do not change these values during an upgrade. They should be set only during installation. Although not impossible, it is inadvisable to change the user under which containers are run, because changing that user involves changing ownership of all the files and folders mounted to or contained within the volume attached to the container. However, if you are required to make this change, see Change the UID for the Postgres Container and Change the UID for the Redis Container for instructions.

```
None
# The UID to be used by TC containers. (Ex. 2000)
THREATCONNECT_UID=
# The GID to be used by TC containers. (Ex. 2000)
THREATCONNECT_GID=
# The TC JOB UID to be used by TC containers. (Ex. 2001)
TC_JOB_UID=
# The TC JOB GID to be used by TC containers. (Ex. 2001)
TC_JOB_GID=
# Database user UID. (Ex. 2002)
DB_UID=
# Database user GID. (Ex. 2002)
DB_GID=
# Redis user UID. (Ex. 2003)
REDIS_UID=
# Redis user GID. (Ex. 2003)
REDIS_GID=
```

Redis Connection Settings Variables

The following environment variables were added to establish a TLS connection to the Redis container:

```
None
# The port for redis [Required]
REDIS_PORT=6379
# Redis data folder [Required] (Ex. /opt/threatconnect-docker/redis-data)
REDIS_DATA=
# Secure Redis. [Optional] [true | false]
REDIS_SECURE=false
# Secure Redis User Name. [Optional]
REDIS_USER=
```

```
# Secure Redis Password. [Optional]
REDIS_PASS=
# Secure Redis TLS. [Optional] [true | false]
REDIS_TLS=false
# Secure Redis Port. [Optional]
REDIS_TLS_PORT=6379
```

Preserve MDB Settings

The following environment variable was added to preserve manual edits to `threatconnect.xml` across restarts:

Note: This variable was added to resolve losses that were occurring during upgrades for legacy operating-system deployments.

None

```
# Run Setup [Optional]. Default: false. Set to true for install or upgrade. Set
to false to persist manual edits to threatconnect.xml over restarts.
TC_RUN_SETUP=true
```

AWS Document Storage

The following environment variables were added to allow AWS document storage instead of local document storage:

None

```
# The AWS access ID [Required if TC_DOCUMENT_STORAGE_TYPE is AWS].
TC_AWS_ACCESS_ID=
# The AWS bucket [Required if TC_DOCUMENT_STORAGE_TYPE is AWS].
TC_AWS_BUCKET_NAME=
# The AWS region [Required if TC_DOCUMENT_STORAGE_TYPE is AWS].
TC_AWS_REGION=
# The AWS secret [Required if TC_DOCUMENT_STORAGE_TYPE is AWS].
TC_AWS_SECRET_KEY=
```

Remove Nginx Container

The nginx container was removed from the configuration. The nginx layer added an unnecessary complexity when running ThreatConnect on multiple hosts. Web traffic logging can now be enabled by setting `TC_ACCESS_LOGGING=true` and captured in a file called `access.log` that resides in the same location as `server.log` and `tc.log` on the host running the `tc-app` container.

Run the following commands to shut down and remove the nginx container and image:

```
None
docker rm -f nginx
docker image list
docker image rm <nginx image id>
```